Schwab

# Programmierung und Bedienung des digitalen Kleinrechenautomaten

# C 8205

## Heft 1

## Grundlagen der Programmierung

## Teil 1

Inhaltsverzeichnis


| | | Seite |
|---|---|---|
| | Vorwort | |
| 0. | Einleitung | |
| 1. | Mathematische Grundbegriffe | 10 |
| 1.1. | Zahlensysteme | 10 |
| 1.1.1. | Positionssysteme der Rechentechnik | 13 |
| 1.1.2. | Dualcodierte Dezimalzahlen | 16 |
| 1.2. | Zahlendarstellung im Rechenautomaten | 19 |
| 1.3. | Algorithmen | 25 |
| 2. | Zahlenumwandlung | 39 |
| 3. | Rechenoperationen im Dualsystem | 46 |
| 3.1. | Duale Addition | 46 |
| 3.2. | Duale Subtraktion | 50 |
| 3.3. | Duale Multiplikation | 58 |
| 3.4. | Duale Division | 59 |
| 3.5. | Rechnen mit dual codierten Zahlen | 59 |
| 3.6. | Die Boolesche Algebra | 62 |
| 4. | Beschreibung einer programmgesteuerten digitalen Rechenanlage | 73 |
| 4.1. | Arbeitsweise digitaler Rechenautomaten | 73 |
| 4.1.1. | Programmsteuerung | 73 |
| 4.1.2. | Befehle | 73 |
| 4.1.3. | Das Maschinenwort | 76 |
| 4.1.4. | Logische Entscheidungen | 76 |
| 4.2. | Struktur programmgesteuerter Rechenautomaten | 77 |
| 5. | Grundbegriffe der Programmierung | 80 |
| 5.1. | Aufgabe der Programmierung | 80 |
| 5.2. | Arbeitsweise bei der Programmierung | 80 |
| 5.3. | Struktur von Programmen | 81 |
| 5.3.1. | Geradeausprogramm und zyklisches Programm | 81 |
| 5.3.2. | Unterprogramm | 84 |
| 5.4. | Adressenänderungen | 86 |
| 5.4.1. | Adressenänderungen 1. Art | 86 |
| 5.4.2. | Adressenänderungen 2. Art | 87 |
| 5.4.3. | Adressenänderungen 3. Art | 88 |

| | | Seite |
|---|---|---|
| 5.5. | Umsetzung allgemeingültiger Algorithmen in die Maschinensprache | 88 |
| 5.5.1. | Manuelle Umsetzung | 88 |
| 5.5.2. | Interpretationssysteme | 89 |
| 5.5.3. | Automatische Umsetzung maschinenunabhängiger algorithmischer Eingangssprachen | 90 |

Anhang

| | |
|---|---|
| A. | TGL Informationsverarbeitung- Datenfluß- und Programmablaufpläne, Sinnbilder |
| B. | Programmbibliothek C 8205 |
| C. | Beschreibungssystematik für Bibliotheksprogramme C 8205 |
| D. | Formale Beschreibung von Sprachen - Normalform von Backus |
| E. | Literaturverzeichnis |

## Vorwort

Heft 1 der Programmierungsanleitung für den digitalen Kleinrechenautomaten C 8205 (D 4a) will den Leser mit den Voraussetzungen und Grundlagen des Programmierens vertraut machen. Auf die Kenntnisse dieses Heftes bauen die folgenden Hefte der Programmierungsanleitung auf.

Nach Einführen einiger Bezeichnungen werden die Zahlensysteme erläutert, die man bei der Arbeit mit dem Cellatron C 8205 benötigt.

Vermittelt werden auch Vorstellungen und Kenntnisse von den Rechenregeln selber, den Algorithmen, mit denen und nach denen die Aufgaben aufbereitet werden, deren Lösung dann dem Automaten übertragen ist.

Zum besseren Verständnis der Arbeit mit dem Automaten sind Kenntnisse über die Lösungsweise der auszuführenden Rechenoperationen im Automaten notwendig.

Im folgenden Teil des Heftes 1 wird eine Einführung in die Programmierungstechnik gegeben.

Der Leser lernt die Struktur und Arbeitsweise von digitalen Rechenanlagen kennen und er wird mit den Grundbegriffen der Programmierung vertraut gemacht.

## 0. Einleitung

Zur Vorbereitung eines Prozesses, beispielsweise eines Produktionsprozesses, plant man das Eintreten spezieller Situationen zum vorgegebenen Zeitpunkt durch qualitative und quantitative Vorgabe eines Sortiments von Merkmalen. Ob sich die Situation in der geplanten Weise realisieren läßt, ist jedoch nicht sicher.

Das Eintreten einer realen Situation (ob man, wie zuvor bemerkt, geplant oder im Verlauf eines beliebigen Prozesses erwartet) ist somit unsicher. Zur Verminderung der Unsicherheit über das Eintreten einer spezifischen realen Situation muß man den Merkmalsbereich zur Charakterisierung der realen Situation möglichst genau bestimmen und die Qualität der jedem Merkmal innewohnenden Aussage ausreichend genau vorgeben, respektive über die Vorgabe hinaus verbessern. Dies erfolgt über Informationen.

Informationen sind also Aussagen über Gegenstand, Erscheinung, Vorgang u. a., die helfen, Unsicherheiten über das Eintreten einer gegenwärtigen oder zukünftigen realen Situation zu verringern oder zu beseitigen. Dabei können die Informationen aus der Auswertung vergangener Zeiträume gewonnen werden.

Rechenautomaten verarbeiten besonders aufbereitete Informationen, sogenannte Daten.

Daten sollen also alle speziell für die automatische Datenverarbeitung aufbereitete Informationen genannt werden. Bei Informationen, und somit auch bei der Spezialform Daten, sind drei Aspekte zu unterscheiden:

- ihre Semantik (oder Bedeutung)
- ihre physikalische Erfassung
- ihre syntaktische Darstellung.

Die Semantik einer Information entspricht der üblichen Bedeutung des Wortes "Information" der Umgangssprache - also eine Auskunft, eine Nachricht. Im Sinne der Datenverarbeitung müssen hierzu qualitative oder quantitative Merkmale und deren Geltungsbereich definiert werden, aus dem dann die Informationen durch Angabe einzelner oder mehrerer Werte für Merkmale ihre Bedeutung erhält.

Die physikalische Erfassung der Information stellt, insbesondere unter dem Aspekt der Datenverarbeitung, hohe Anforderungen an die Auswahl aus der Vielfalt der Möglichkeiten. Vorwiegend technische Begrenzungen lassen es als zweckmäßig erscheinen, solche physikalische Darstellung zu wählen, die durch bistabile Bausteine realisiert werden kann. Das läßt sich ermöglichen, beispielsweise:

| | |
|---|---|
| Impuls | - Fehlen eines Impulses |
| Lochung | - Fehlen einer Lochung |
| lokal polarisiertes Magnetfeld | - entgegengerichtetes lokal polarisiertes Magnetfeld |

positiver Remanenzzustand – negativer Remanenzzustand eines Ferritkernes

u. a. Hierdurch entstehen Signale als physikalische Träger einer Information.

Dieser Sachverhalt läßt sich auch leicht logisch über eine zweiwertige Logik erfassen, deren Elemente lediglich den Wahrheitswert "wahr" oder "falsch" annehmen können. Ersetzt man diesen Wahrheitswert in üblicher Weise durch Ziffern, beispielsweise

1 = "wahr"
0 = "falsch" ,

so hat man die übliche Darstellung logischer Variabler, deren Wertevorrat eben nur "0" und "1", letztere auch vielfach "L" geschrieben, ist. Da das Signal eines bistabilen Bausteines physikalischer Träger einer Information ist, diese wiederum durch logische Variablen einer zweiwertigen Logik theoretisch erfaßt werden können, bilden diese Variablen letzthin die Elementarzeichen einer Information.

Andererseits wird im folgenden ein besonders einfaches Zahlensystem, das Dualsystem, beschrieben, das ebenfalls nur aus den beiden Ziffern "0" und "L" besteht. Es liegt nun nahe, den Wertevorrat logistischer Variablen durch Dualziffern auszudrücken, was praktisch möglich ist. Damit ergibt sich, daß die Dualziffer

- Darstellungszeichen eines Zustandes bistabiler Elemente
- Argument der Variablen einer zweiwertigen Logik und
- Ziffer des dualen Zahlensystems

sein kann. Dann lassen sich Erkenntnisse aus dem einen Bereich sinnvoll in den anderen übertragen.

Die Dualziffer wird als Abkürzung der entsprechenden englischen Bezeichnung "binary digit" - auch Bit - genannt. Daher kann gesagt werden:

Das Bit ist die Elementareinheit der Information. Es ist besonders geeignet, die physikalische Erfassung der Information als Signal zu ermöglichen.

Der Syntax gibt an, wie sich die Informationen und damit die Daten aus Elementareinheiten in Form von

- Zeichen
- Worten
- Sätzen
- Gruppen
- Blöcken

aufbauen lassen.

Das Zeichen ist ein Element aus einer zu vereinbarenden Menge von Elementen. Für die Rechentechnik werden die Zeichen so vereinbart, daß sie einmal nach bestimmten Regeln aus Elementareinheiten aufgebaut (diese Regeln nennt man Code, auf den später genauer eingegangen wird) und damit technisch realisiert werden können, zum anderen aber auch unmittelbar manuell lesbar sind (ohne auf Elementareinheiten zurückgehen zu müssen). In der Rechentechnik haben drei Zeichenarten Bedeutung:

- numerische Zeichen, hier besteht die Menge der Zeichen aus den Ziffern 0 bis 9

  Z = 0, 1, 2, ... , 9

- alphabetische Zeichen, deren Zeichenvorrat aus den Buchstaben des großen lateinischen Grundalphabets besteht, also

  Z = A, B, C, ... , Z und

- Sonderzeichen. Bei ihnen bildet man ein für jede Anlage unterschiedliches Sortiment, beispielsweise

  +, -, x, /, ., ;, (, ), =, ', ",

  und andere. Ggf. wird auch das kleine lateinische Alphabet oder das Alphabet einer anderen Sprache (kyrillisches Alphabet) verwendet.

Nach den verwendeten Zeichen unterscheidet man zwei Arbeitsweisen in der Datenverarbeitung:

- Numerische Arbeitsweise

  bei ihr verwendet man numerische und Sonderzeichen und

- Alphanumerische Arbeitsweise

  bei ihr verwendet man, wie der Name bereits andeutet, alphabetische, numerische und Sonderzeichen.

Das Wort ist eine Folge von Zeichen. Insbesondere bezeichnet man als Maschinenwort eine solche Zeichenfolge, die von einem Automaten gelesen und interpretiert werden kann. Je nach Automatentyp kann man mit

- Wörtern fester Länge (für sie ist eine feste Anzahl von Bits -beim Cellatron C 8205 sind es 33 Bit- oder Zeichen als Wortlänge vorgegeben) und/oder
- Wörter variabler Länge (bei ihnen wird nur eine Maximallänge vorgeschrieben)

arbeiten.

In der Datenverarbeitung unterscheidet man zwei Wortarten: Befehlswörter und Zahlwörter.

Befehlswörter ermöglichen die Darstellung eines Befehls (auf diesen wird später eingegangen).

Zahlwörter dienen zur Zahlendarstellung in der DVA (Datenverarbeitungsanlage). Dabei unterscheidet man:

- Festkommazahlen
- Gleitkommazahlen
- Ganzzahlen
- Zahlen mit programmierbarer Kommastellung u. a.

Hierauf wird im Abschnitt 1.2. ausführlich eingegangen.

Sätze, insbesondere Datensätze, werden aus einem oder mehreren gleichen oder unterschiedlichen Wörtern zusammengesetzt und durch eine sogenannte Satzmarke abgeschlossen.

Gruppen nennt man die Zusammenfassung mehrerer zusammengehöriger Datensätze. (Ein Zwischenschalten von Datengruppen kann jedoch auch unterbleiben.)

Blöcke ist schließlich die Bezeichnung der komplexen Einheiten für zusammengehörige Datensätze oder Gruppen.

Datei oder Datenbank wird die Gesamtheit der zu einem Begriff gehörenden Datenblöcke, evtl. unter Einbeziehung einzelner Gruppen oder Sätze, genannt.
Die Regeln und Vorschriften zur Bildung der Daten-Struktureinheiten liefern den Syntax - die Grammatik.

Von besonderer Bedeutung für das Programmieren und das Bedienen eines Automaten ist die Kenntnis der automateninternen Darstellung der Zeichen. Sie erfolgt aus technischen Gründen, wie bereits ausgeführt, in Dualdarstellung. Die Umwandlungsvorschriften von Zeichen in/aus Dualdarstellung nennt man Codes.

Der Code ist vom Rechenautomaten abhängig. Zur Darstellung der numerischen Zeichen 0,...,9 benötigt man mindestens 4 Bit. Jedes Bit eines Zeichens kann beim Transport im Automaten gleichzeitig durch einen gesonderten Kanal gegeben werden (zeichenserielle Übertragung), daher spricht man auch von einem 4-Kanal-Code. Wird beispielsweise folgende Zuordnung gefordert:

| | |
|---|---|
| OOOO | 0 |
| OOOL | 1 |
| OOLO | 2 |
| OOLL | 3 |
| OLOO | 4 |
| OLOL | 5 |
| OLLO | 6 |
| OLLL | 7 |
| LOOO | 8 |
| LOOL | 9 |

LOLO Pseudodezimale
LOLL für Sonderzeichen
LLOO
LLOL
LLLO
LLLL

dann liegt der sogenannte Direktcode vor, auf den später nochmals eingegangen wird. Alle nicht durch Ziffern verbrauchten Kombinationen nennt man Pseudodezimale. Sie können zur Darstellung von Sonderzeichen genutzt werden.

Mit diesen wenigen Zeichen kommt man jedoch in der modernen Datenverarbeitung nicht aus. Andererseits lassen sich nur die in nachfolgender Aufstellung angegebene Anzahl von Zeichen aus zwei Dualziffern, bei vorgegebener Anzahl der Dualstellen des Zeichens, aufstellen:

1 Dualstelle = $2^1$ Zeichen = 2
2 Dualstellen= $2^2$ Zeichen = 4
3 Dualstellen= $2^3$ Zeichen = 8
4 Dualstellen= $2^4$ Zeichen = 16
5 Dualstellen= $2^5$ Zeichen = 32
6 Dualstellen= $2^6$ Zeichen = 64
7 Dualstellen= $2^7$ Zeichen = 128
8 Dualstellen= $2^8$ Zeichen = 256

In der Praxis haben sich

Fünf -Kanal-Codes
Sechs -Kanal-Codes
Sieben -Kanal-Codes
Acht -Kanal-Codes

durchgesetzt. Meist jedoch verwendet man den Fünf- und den Acht-Kanal-Code. Für den Cellatron C 8205 gilt u. a. der Fünf-Kanal-Code des internationalen Telegrafenalphabets Nr. 2 mit folgender Festlegung:

| | | | | | |
|---|---|---|---|---|---|
| 00 000 | Leerzeichen | | LO 000 | E | 3 |
| 00 00L | T | 5 | LO 00L | Z | + |
| 00 0LO | Wagenrücklauf | | LO 0LO | D | Werda |
| 00 0LL | O | 9 | LO 0LL | B | ? |
| 00 L00 | Zwischenraum | | LO L00 | S | , |
| 00 L0L | H | | LO L0L | Y | 6 |
| 00 LL0 | N | | LO LL0 | F | |
| 00 LLL | M | | LO LLL | X | / |
| 0L 000 | Zeilenvorschub | | LL 000 | A | - |
| 0L 00L | L | ) | LL 00L | W | 2 |
| 0L 0L0 | R | 4 | LL 0L0 | J | Kl |
| 0L 0LL | G | | LL 0LL | Ziffern und Zeichen | |
| 0L L00 | I | 8 | LL L00 | U | 7 |
| 0L L0L | P | 0 | LL L0L | Q | 1 |
| 0L LL0 | C | : | LL LL0 | K | ( |
| 0L LLL | V | = | LL LLL | Buchstaben | |

Bei der Datenübertragung und -verarbeitung können durch Störeinwirkungen (Ausfall einzelner Elemente) Verfälschungen der Daten auftreten. Das muß weitestgehend vermieden werden. Systematisch auftretende Fehler können durch geeignete Teste leicht gefunden und beseitigt werden. Zufällig auftretende Fehler werden durch sogenannte Prüfbitkontrolle stark reduziert. Je nach Automatentyp ist für ein Zeichen oder ein Wort eine gerade oder ungerade Anzahl von Dualzeichen L vorgeschrieben. Es soll im folgenden angenommen werden, daß eine ungerade Anzahl von Werten L festgelegt sei. Dann wird ein besonderes Bit benutzt, um bei der Eingabe in den Automaten immer dieser Forderung zu genügen. Hat ein Zeichen oder ein Wort ohne Berücksichtigung des besonderen Bits, auch Prüfbit genannt, eine gerade Anzahl Dualzeichen L, so ist das Prüfbit L, anderenfalls bei einer ungeraden Anzahl Dualzeichen L, ist das Prüfbit O.

Die interne Arbeitsweise nahezu jedes Automaten sichert, daß die ungerade Anzahl der Dualzeichen L je Zeichen oder Wort bei der normalen Arbeit (arithmetische Operationen, Transport u. a.) immer erhalten bleibt. Fällt durch Störung ein Einzelbit aus oder wird es unberechtigt

erzeugt, wird die Anzahl der Werte L im Zeichen oder Wort gerade, und bei der nächsten Kontrolle geht der Automat in Stopp und zeigt den Fehler an. Dadurch wird der Ausfall oder das Hinzufügen eines Bits nahezu völlig ausgeschaltet . Daß in einem Zeichen oder Wort gleichzeitig zwei Bit ausfallen oder hinzukommen, ist unwahrscheinlich und kann gegebenenfalls durch Rechenkontrollen erkannt werden.

Die Prüfbitkontrolle kann sich auf ein Zeichen beziehen, wie beim Acht-Kanal-Code des R 300. Hier haben die Bits eines Zeichens (dessen Dualdarstellung insgesamt auch als Byte bezeichnet wird) folgende Bedeutung:

| Bit | Wert | Bezeichnung |
|---|---|---|
| 8 | W | Wortmarkenbit |
| 7 | V | Überbit V |
| 6 | U | Überbit U |
| 5 | Pr | Prüfbit |
| 4 | $2^4$ | numerisches Bit der Bewertung 8 |
| 3 | $2^3$ | numerisches Bit der Bewertung 4 |
| 2 | $2^1$ | numerisches Bit der Bewertung 2 |
| 1 | $2^0$ | numerisches Bit der Bewertung 1 |

Die Bedeutung der Dualkombinationen ist aus nachfolgender Tabelle ersichtlich.

| Num. Zone / UV | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ✖ | ÷ | ( | : | ▽ |
| 0L | | + | A | B | C | D | E | F | G | H | I | ~ | . | ; | ! | ⏻ |
| L0 | | - | J | K | L | M | N | O | P | Q | R | ≈ | ) | * | = | < |
| LL | | 9 | / | S | T | U | V | W | X | Y | Z | ≋ | , | % | △ | > |

Das Prüfbit hat die bereits angeführte Bedeutung, das Wortmarkenbit W kennzeichnet das Wortende.

✖ Leerzeichen
9 Wortmarkenkennzeichen extern
~ Satzmarke
≈ Gruppenmarke

Blockmarke

Blockanfang 1 auf Magnetband

Blockanfang 2 auf Magnetband

Wortmarke auf Magnetband

 Stanzverhinderungszeichen

Die Prüfbitkontrolle kann sich aber auch auf ein ganzes Wort beziehen, denn die Prüfung der Dualdarstellung der einzelnen Zeichen erfordert einen großen Aufwand. Die Sicherheit ist jedoch ausreichend, wenn bei fester Wortlänge nur eine Prüfbitkontrolle des Wortes vorgenommen wird.

## 1. Mathematische Grundbegriffe

Trotz alphanumerischer Arbeitsweise vollzieht sich die eigentliche Datenverarbeitung rein numerisch. Die zu verarbeitenden Informationen werden daher als Zahlen dargestellt. Es müssen also geeignete Zahlensysteme zur Datenverarbeitung bereitgestellt werden.

### 1.1. Zahlensysteme

An Zahlensysteme werden bestimmte Anforderungen gestellt:

- Die Zahlen müssen sich in einem Zahlensystem schnell und bequem schreiben lassen.
- Das Zahlensystem und seine Bezeichnungsweise müssen leicht erlernbar sein.
- Es muß, zumindest theoretisch, jede noch so große und auch jede noch so kleine reelle Zahl darstellbar sein.
- Man muß mit den Zahlen leicht rechnen können.

Einfachste Formen ergeben sich bereits durch Bündeln. Jede Einheit wird durch einen Gegenstand markiert, z.B. durch einen Strich oder einen Stein. Zur besseren Übersicht werden beispielsweise vier Striche durch einen fünften durchstrichen oder alle Steinchen zu 10 gehäuft.

Als Bündelungswert kann jede Zahl, größer als 1, genommen werden. Praktisch haben sich jedoch die Zahlen 5 und 10 durchgesetzt, weil man bei derartiger Bündelung mit den Fingern (digits) auch gleich ein Rechenhilfsmittel (eine Rechenmaschine) zur Verfügung hat.

Im Laufe einer langwierigen historischen Entwicklung haben sich dann aus einer primitiven Bündelung Zahlensysteme entwickelt. Bei modernen Zahlensystemen wird eine Einheit und eine Basiszahl vorgegeben. Wächst die Anzahl der Einheiten soweit an, daß sie der Basiszahl entsprechen, werden sie "gebündelt" und er-

geben eine Einheit des nächsthöheren Stellenwertes usw. Dabei häuft man die Einheiten nicht mehr an, sondern ordnet jeder Anzahl ein spezielles Zeichen, Ziffer genannt, zu. Auf diese Weise erhält man ein Positionssystem.

Das Positionssystem wird durch folgende Forderungen charakterisiert:

0. Gegeben sei eine Einheit 1 des Zahlensystems.
1. Gegeben sei eine ganze positive Zahl, größer als 1, nach der gebündelt wird. Diese Zahl soll Basiszahl heißen und mit B bezeichnet werden.
2. Dann werden B-1 Ziffern und die Null benötigt, was wie folgt dargestellt werden kann:

   $z_i = (0, 1, 2, \ldots, (B - 1))$

   (i soll einen Index andeuten, der später zur Zahlenschreibweise benötigt wird.)
3. Jede Ziffer erhält entsprechend der Bezeichnung einen Ziffernwert.
4. Die Zahl ergibt sich dann wie folgt:
   Für jede Stelle wird der Ziffernwert mit einer Potenz der Basiszahl (dem Stellenwert) multipliziert

   $z_i \cdot B^i$

   Es wird über alle Produkte summiert,

$$Z = \ldots + z_n \cdot B^n + \ldots z_1 \cdot B^1 + z_0 \cdot B^0 + z_1 \cdot B^{-1} + \ldots + z_{-m} \cdot B^{-m} + \ldots$$

$$= \sum_i z_i \cdot B^i$$

   Dabei können die $z_i$ nur Ziffern sein, also nur die Werte 0, 1, 2, ... , (B -1) annehmen. Der Index i durchläuft den Wertebereich

$$i = + \ , \ldots , n, \ldots , 1, 0, -1, \ldots , -m, \ldots , -$$

Die endlichen Zahlen ergeben sich dann mit

$z_i = 0$ für $i = n+1, n+2, \ldots, +\infty$ und

$i = -(m+1), -(m+2), \ldots, -\infty$

Exponent der Basispotenz und Index stimmen überein. Wenn also im nachfolgenden Abschnitt 5 die Potenzen der Basiszahlen nicht mehr mitgeschrieben werden, sind durch Abzählen des Index vom Komma nach rechts oder links die jeweils zugehörigen Stellenwerte rekonstruierbar.

5. Für die Schreibweise wird vereinbart, daß die Potenzen von B und das Additionszeichen nicht mitgeschrieben werden.

$$Z = z_n \ldots z_1 z_0 , z_{-1} \ldots z_{-m}$$

Zudem wird die Ziffernfolge mit nichtnegativer Basispotenz von der mit negativer Potenz der Basis durch ein Interpunktionszeichen (meist Komma) getrennt. Dadurch erhält jede Ziffer ihre Stelle in der Zahl entsprechend dem Stellenwert.
Vom Komma nach links ist der Stellenwert für die

1. Stelle = $B^0$ ($10^0 = 1$)
2. Stelle = $B^1$ ($10^1 = 10$)
3. Stelle = $B^2$ ($10^2 = 100$)
:
n-te Stelle = $B^{n-1}$ ($10^{n-1}$)

und nach rechts für die

1. Stelle nach dem Komma = $B^{-1}$ ($10^{-1} = 0,1$)
2. Stelle nach dem Komma = $B^{-2}$ ($10^{-2} = 0,01$)
3. Stelle nach dem Komma = $B^{-3}$ ($10^{-3} = 0,001$)
:
m-te Stelle nach dem Komma = $B^{-m}$ ($10^{-m}$)

(Die Stellenwerte des Zahlensystems wurden zur Übersicht angefügt.)

Theoretisch gibt es unendlich viele Positionssysteme. Alle natürlichen Zahlen, die größer als 1 sind, können ja als Basiszahl genommen werden. Praktisch jedoch, insbesondere in der maschinellen Rechentechnik – speziell für den Cellatron C 8205- sind drei Systeme von Bedeutung:

- das Dezimalsystem
- das Oktalsystem
- das Dual- oder Binärsystem.

### 1.1.1. Positionssysteme der Rechentechnik

Die allgemeinen Forderungen an ein Positionssystem sollen nunmehr auf die genannten Systeme präzisiert werden.

0. Für alle Positionssysteme, somit für das Dezimal-, Oktal- und Dualsystem, ist die Einheit 1.
1. Die genannten Zahlensysteme haben folgende Basiszahlen:

   Dezimalsystem B = 10
   Oktalsystem B = 8
   Dualsystem B = 2

   Dabei ist besonders für das spätere Umrechnen bedeutungsvoll, daß $8 = 2^3$ ist. Leider ist die Basiszahl des Dezimalsystems keine Potenz der Basiszahlen der beiden anderen Systeme, woraus einige Schwierigkeiten der Umformung entstehen werden.
2. Entsprechend den angeführten Basiszahlen werden folgende Ziffern $z_i$ benötigt:

   $B = 2 \quad z_i = \{0,1\} = \{0,L\}$
   $B = 8 \quad z_i = \{0,1,2,3,4,5,6,7\}$
   $B = 10 \quad z_i = \{0,1,2,3,4,5,6,7,8,9\}$

   Im Dualsystem ist es üblich, die duale Eins durch das Zeichen L darzustellen.

Eine Dualziffer bezeichnet man auch als Bit (binary digit - engl. - Binärstelle [1]).

3. Den Ziffernwert erhält man, auf das Dezimalsystem bezogen, zu:

| B = | 10 | 8 | 2 | | |
|---|---|---|---|---|---|
| | 0 | 0 | 0 | = | null Einheiten |
| | 1 | 1 | L | = | eine Einheit |
| | 2 | 2 | | = | zwei Einheiten |
| | 3 | 3 | | = | drei Einheiten |
| | 4 | 4 | | = | vier Einheiten |
| | 5 | 5 | | = | fünf Einheiten |
| | 6 | 6 | | = | sechs Einheiten |
| | 7 | 7 | | = | sieben Einheiten |
| | 8 | | | = | acht Einheiten |
| | 9 | | | = | neun Einheiten |

4. Die Zahlen erhält man wie beschrieben durch Multiplizieren der Ziffer mit dem Stellenwert und Summation über alle vorhandenen von Null verschiedenen Ziffern. Also für:

B = 10 zu
$Z = z_n \cdot 10^n + z_{n-1} \cdot 10^{n-1} + \ldots z_0 \cdot 10^0 + \ldots + z_m \cdot 10^{-m}$
B = 8 zu
$Z = z_n \cdot 8^n + z_{n-1} \cdot 8^{n-1} + \ldots z_0 \cdot 8^0 + \ldots + z_{-m} \cdot 8^{-m}$
B = 2 zu
$Z = z_n \cdot 2^n + z_{n-1} \cdot 2^{n-1} + \ldots z_0 \cdot 2^0 + \ldots + z_{-m} \cdot 2^{-m}$

wobei die $z_i$ nur Werte der für das jeweilige Zahlensystem in 2. definierten Ziffern annehmen dürfen.

5. Für die Zahlendarstellung sei zusätzlich vereinbart, daß bei dem weniger bekannten Oktalsystem die Zahlen in Klammern geschrieben und die Basis als Index hinzugefügt wird. So haben alle nachfolgend angeführten Zahlen den gleichen Wert.

4608, 06640625
$(1\ 1000, 042)_8$
L 00L 000 000 000, 000 L00 0L

[1]) binär: aus zwei Ziffern bestehend

Der Stellenwert ist natürlich in jedem Zahlensystem verschieden und ergibt sich für

B = 10

4. Stelle vor dem Komma = $10^3$ = 1 000
3. Stelle vor dem Komma = $10^2$ = 100
2. Stelle vor dem Komma = $10^1$ = 10
1. Stelle vor dem Komma = $10^0$ = 1
1. Stelle nach dem Komma = $10^{-1}$ = 0,1
2. Stelle nach dem Komma = $10^{-2}$ = 0,01
3. Stelle nach dem Komma = $10^{-3}$ = 0,001

B = 8

4. Stelle vor dem Komma = $8^3$ = (1 000 )$_8$= 512
3. Stelle vor dem Komma = $8^2$ = ( 100 )$_8$= 64
2. Stelle vor dem Komma = $8^1$ = ( 10 )$_8$= 8
1. Stelle vor dem Komma = $8^0$ = ( 1 )$_8$= 1
1. Stelle nach dem Komma = $8^{-1}$ = ( 0,1 )$_8$=0,125
2. Stelle nach dem Komma = $8^{-2}$ = ( 0,01 )$_8$=0,015625
3. Stelle nach dem Komma = $8^{-3}$ = ( 0,001)$_8$=0,0019531

B = 2

4. Stelle vor dem Komma = $2^3$ = L 000 = 8
3. Stelle vor dem Komma = $2^2$ = L00 = 4
2. Stelle vor dem Komma = $2^1$ = L0 = 2
1. Stelle vor dem Komma = $2^0$ = 0 = 1
1. Stelle nach dem Komma = $2^{-1}$ = 0,L = 0,5
2. Stelle nach dem Komma = $2^{-2}$ = 0,0L = 0,25
3. Stelle nach dem Komma = $2^{-3}$ = 0,00L = 0,125

Im Dualsystem können jeweils drei Dualstellen zu einer Dualtriade zusammengefaßt werden.
Da wir die Stellenwerte für das Umformen der Zahlen häufiger benötigen, sind sie auf einer Beilage zusammengestellt.

Wie aus diesem Vergleich bereits ersichtlich, hat insbesondere der ganzzahlige Anteil der Dualzahlen eine weitaus größere Stellenzahl als der entsprechende Anteil der Dezimalzahlen. Dies ist ein we-

sentlicher Nachteil der Dualzahlen, denn deren "Länge" vergrößert sich rund auf das $3\frac{1}{3}$-fache der gleich großen Dezimalzahl. Auch sind Dualzahlen zumindest zu Anfang schwer zu lesen und eben ungewohnt.

In der maschinellen Rechentechnik ist das Dualsystem jedoch von besonderer Bedeutung. Das Rechnen im Dualsystem ist sehr einfach. Es läßt sich, wie später nachgewiesen wird, durch Schaltelemente mit zwei stabilen Zuständen nachbilden und ausführen. Darauf basiert die Technik moderner digitaler Rechenautomaten.

### *.1.2. Dualcodierte Dezimalzahlen

Die Bedeutung des Dezimalsystems in der Praxis und das Gewöhnen an dieses Zahlensystem waren der Anlaß, daß man Dezimalzahlen direkt durch Dualzeichen darstellt. Das ist ebenfalls ein Code, wie bereits im Abschnitt 0. besprochen, nämlich der Dualcode für Dezimalzahlen.

Zur Darstellung der größten Dezimalziffer 9 benötigt man mindestens vier Bit, denn es ist

$$9 = 1001$$

Daher wird erst einmal versucht, Codierungsschlüssel zu finden, bei denen jede Dezimalziffer in eine Dualtetrade übersetzt wird.

Für diese Codierung haben sich in der Rechentechnik drei Schlüssel bewährt.

1. Direkte dezimal-duale Verschlüsselung

   Bei der direkten dezimal-dualen Verschlüsselung, auch Direkt-Code genannt, wird jede Dezimalziffer in die ihrem Ziffernwert entsprechende Dualtetrade verwandelt.

| | | | |
|---|---|---|---|
| 0 = | 0000 | 5 = | 0L0L |
| 1 = | 000L | 6 = | 0LL0 |
| 2 = | 00L0 | 7 = | 0LLL |
| 3 = | 00LL | 8 = | L000 |
| 4 = | 0L00 | 9 = | L00L |

Beispiel: 320,47 = 00LL 00L0 0000, 0L00 L L

Dieser Code hat den Vorteil, daß man die Verschlüsselung leicht lesen kann.

2. Aiken-Verschlüsselung

Wie später noch gezeigt wird, spielt beim Dezimalsystem das sogenannte Neunerkomplement in der maschinellen Rechentechnik eine große Rolle. Unter Neunerkomplement einer Dezimalziffer z soll dabei diejenige Ziffer z' verstanden werden, die zur ursprünglichen addiert 9 ergibt.
Also

$$z + z' = 9$$

Damit das Neunerkomplement leicht gebildet werden kann, hat der Amerikaner Aiken folgenden Code eingeführt:

| | | | |
|---|---|---|---|
| 0 = | 0000 | 9 = | LLLL |
| 1 = | 000L | 8 = | LLL0 |
| 2 = | 00L0 | 7 = | LLOL |
| 3 = | 00LL | 6 = | LL00 |
| 4 = | 0L00 | 5 = | L0LL |

Beispiel: 320,47 = 00LL 00L0 0000, 0L00 LL0L

Die Neunerkomplemente sind nebeneinander geschrieben. Die entsprechenden Dualtetraden unterscheiden sich dabei nur darin, daß beim Komplement jeweils für eine 0 ein L und umgekehrt für ein L eine 0 gesetzt ist. Diese Operation nennt man Negation. Das Neunerkomplement wird beim Aiken-Code somit durch Negation der Dualtetrade gebildet.

3. Dreierexzeß-Verschlüsselung

Ein weiterer Code, aus dem das Neunerkomplement durch Negation der Dualtetraden ermittelt werden kann, ist der Dreierexzeß-Code. Er entsteht dadurch, daß zur Dezimalziffer z drei addiert und dann die zugehörige Dualtetrade bestimmt wird.

| | | | | | |
|---|---|---|---|---|---|
| 0 | (+ 3 = 3) | = OOLL | 9 | (+ 3 = 12) | = LLOO |
| 1 | (+ 3 = 4) | = OLOO | 8 | (+ 3 = 11) | = LOLL |
| 2 | (+ 3 = 5) | = OLOL | 7 | (+ 3 = 10) | = LOLO |
| 3 | (+ 3 = 6) | = OLLO | 6 | (+ 3 = 9) | = LOOL |
| 4 | (+ 3 = 7) | = OLLL | 5 | (+ 3 = 8) | = LOOO |

Beispiel: 320,47 = OLLO OLOL OOLL, OLLL LOLO

Der Vorteil der Dreierexzeß-Verschlüsselung besteht vor allem darin, daß in jeder Dualtetrade beide Dualziffern O und L vorkommen. Hierdurch können automatische Kontrollen leichter ausgeführt werden.

Es lassen sich noch viele Verschlüsselungssysteme dieser Art aufstellen, da aus den Ziffern O und L insgesamt 16 Tetraden gebildet werden können. Praktisch haben aber nur die oben erwähnten Bedeutung, die nochmals zusammengefaßt werden sollen:

| | Direkt-Code | Aiken-Code | Dreierexzeß-Code |
|---|---|---|---|
| OOOO | 0 | 0 | - |
| OOOL | 1 | 1 | - |
| OOLO | 2 | 2 | - |
| OOLL | 3 | 3 | 0 |
| OLOO | 4 | 4 | 1 |
| OLOL | 5 | - | 2 |
| OLLO | 6 | - | 3 |
| OLLL | 7 | - | 4 |
| LOOO | 8 | - | 5 |

| | Direkt-Code | Aiken-Code | Dreierexzeß-Code |
|---|---|---|---|
| LOOL | 9 | - | 6 |
| LOLO | - | - | 7 |
| LOLL | - | 5 | 8 |
| LLOO | - | 6 | 9 |
| LLOL | - | 7 | - |
| LLLO | - | 8 | - |
| LLLL | - | 9 | - |

Die Dualtetraden, denen in einem Code keine Dezimalziffer entspricht (durch Strich markiert), werden Pseudotetraden genannt.

Pseudotetraden können dazu verwendet werden, um beispielsweise bei rein numerischer Verschlüsselung einen Code für die Sonderzeichen festzulegen.

## 1.2. Zahlendarstellung im Rechenautomaten

Die Forderung an Zahlensysteme, "jede auch noch so große - und entsprechend auch noch so kleine Zahl erfassen zu können", stößt bei der automateninternen Darstellung der Zahlen auf Schwierigkeiten. Gleich, ob konstante oder variable Wortlänge, ist die Stellenzahl eines Wortes, insbesondere also auch des Zahlwortes, begrenzt. Beim Cellatron C 8205 ist die konstante Wortlänge 33 Bits.

Diese Begrenzung stellt eine aus technischen Gründen notwendige Einschränkung zur oben fixierten Forderung an Zahlensysteme dar.

Zum kompletten Zahlwort gehört:

1. der absolute Betrag der Zahl (die Ziffernfolge ohne Berücksichtigung des Vorzeichens)
2. das Vorzeichen

3. eventuell Informationen über die Kommastellung

4. Sonderinformationen, beispielsweise die Möglichkeit, eine Zahl besonders zu kennzeichnen.

Um dennoch einen großen Zahlenbereich mit größtmöglicher Geschwindigkeit verarbeiten zu können, werden verschiedene Zahlendarstellungen im Rechenautomaten verwendet, beispielsweise die Darstellung als

- Festkommazahlen
- Zahl mit programmierbarer Kommastellung
- Gleitkommazahlen u. a.

Für die folgende Betrachtung soll die Wortlänge des Cellatron C 8205 mit 33 Bits, dem entsprechen 11 Oktalstellen, zugrunde gelegt werden.

Bei der Festkommadarstellung wird, wie es der Name bereits sagt, eine feste Kommastellung vorgegeben. So gibt es Automaten, die nur mit ganzen Zahlen operieren. Das Komma steht dann hinter der letzten Stelle. Hier ist der Zahlenbereich:

$$0 \leq |Z| \leq (777\ 777\ 777\ 77)_8,$$

wenn 11 Oktalstellen angenommen werden.

Andere Automaten rechnen nur mit Zahlen, die kleiner als 1 sind. Bei diesen Automaten setzt man das Komma vor die erste Stelle. Der Zahlenbereich ergibt sich dann zu:

$$0 \leq |Z| \leq (0{,}777\ 777\ 777\ 77)_8,$$

und die kleinste Schrittweite zwischen zwei aufeinanderfolgenden Zahlen ist:

$$(0,\ 000\ 000\ 000\ 01)_8$$

Ergibt sich bei der Festkommadarstellung im Laufe der Rechnung irgendein Wert, der die oberste Grenze überschreitet, so unterbricht der Automat den Re-

chenprozeß und zeigt einen "Überlauf" an. Das ist eine wichtige Kontrolle, die von Automaten selbständig ausgeführt wird. Wäre diese Kontrolle im logischen Aufbau nicht vorgesehen, gingen die "überlaufenden" Stellen verloren, und das Ergebnis würde verfälscht werden.

Die Festkommadarstellung stellt große Anforderungen an den Programmierer. Er muß die gesamte Rechnung wertmäßig verfolgen und erreichen, daß an keiner Stelle des Rechenablaufes der zulässige Zahlenbereich überschritten wird. Das kann er erzwingen, indem er für die Ausgangswerte der Rechnung entsprechende Einschränkungen vorschreibt. Dabei müssen die Ausgangswerte ggf. mit einem sogenannten Skalenfaktor multipliziert werden, damit sie in den zulässigen Bereich fallen.

Da diese Berechnungen einige Schwierigkeiten bereiten, werden neben der Festkommadarstellung auch die Darstellung der Zahlen mit programmierbarem Komma oder die Gleitkommadarstellung verwandt. Hier soll noch näher auf letztere eingegangen werden.

Will man sehr große oder sehr kleine Zahlen darstellen, so benutzt man eine abkürzende Schreibweise.

Beispielsweise gibt die Loschmidtsche Konstante Auskunft über die Anzahl der Moleküle einer Volumeneinheit. Ihr Wert ist:

$$N_L = 6{,}023 \cdot 10^{23}\ \mathrm{Mol}^{-1}$$

Als anderes Beispiel wird das Plancksche Wirkungsquantum durch den Wert:

$$h = 6{,}625 \cdot 10^{-27}\ \mathrm{erg} \cdot \mathrm{s}$$

angegeben.

Es wurde also die große, respektive sehr kleine

Zahl dadurch ersetzt, daß man eine Zehnerpotenz als Faktor heranzog:

602 300 000 000 000 000 000 000 $= 6{,}023 \cdot 10^{23}$

0, 000 000 000 000 000 000 000 000 006 625 $= 6{,}625 \cdot 10^{-27}$

Diese Möglichkeit der Zahlendarstellung wird auch für die Rechentechnik zunutze, indem man etwas abweichend von der obigen Schreibweise festlegt:

Jede Zahl wird so verschoben, daß sie in den Bereich $0{,}1 \leq |Z| < 1$ fällt. Diesen Vorgang nennt man Normieren. Für diese Bereiche können auch andere Festlegungen getroffen werden, doch ist das weniger gebräuchlich.

Entsprechend der angeführten Normierung ergibt sich dann der Exponent der Zehnerpotenz.

Beispiel: 324,76 wird zu $0{,}32476 \cdot 10^{3}$

0,0247 wird zu $0{,}247 \cdot 10^{-1}$

Wie ersichtlich, ist die Normierung recht einfach auszuführen. Das Komma wird nach links (positiver Exponent) oder nach rechts (negativer Exponent) so verschoben, bis die erste Stelle rechts vom Komma von Null verschieden ist, während alle Stellen links vom Komma Null sind.

Aus der Anzahl der Schritte, um die verschoben wird, ergibt sich dann der absolute Betrag des Exponenten.

Eine solche Darstellung der Zahlen durch:

$$Z = m \cdot B^{e}$$

nennt man halblogarithmisch. Dabei ist

- m die Mantisse
- B die Basis des Zahlensystems
- e der Exponent
- Z die Zahl

Für die Darstellung im Rechenautomaten kann man die Zahl noch vereinfachen, indem man die Null vor dem Komma, das Komma selber und die Basis des Zahlensystems nicht schreibt.

Beispiel: $+ 324.76 = + 0{,}32476 \cdot 10^{+3} = + 32476 + 3$

$+ 0{,}0247 = + 0{,}247 \cdot 10^{-1} = + 24700 - 1$

$- 53428 = - 0{,}53428 \cdot 10^{+5} = - 53428 + 5$

$- 0{,}0003 = - 0{,}3 \cdot 10^{-3} = - 30000 - 3$

Die Null muß zusätzlich festgelegt werden.
Für sie gilt:

$$z = m \cdot B^e$$

mit $m = 0$ und $e = 0$

Durch die Gleitkommadarstellung wird der zulässige Zahlenbereich eines Automaten wesentlich vergrößert.

Für den Cellatron C 8205 ergibt sich:

$$2^{-33} \leq |z| \leq (1 - 2^{-25}) \cdot 2^{31}$$

und $z = 0$

Es ist offensichtlich, daß die Programmierer bei der Gleitkommadarstellung wesentlich weniger auf den zulässigen Zahlenbereich zu achten brauchen, da er seltener überschritten wird. Aber auch in diesem Fall stoppt der Automat die Arbeit bei Überlauf. Für diesen Vorteil nimmt man gern den Nachteil in Kauf, daß die Zahlen vor der Bearbeitung und nach der Berechnung umgeformt werden müssen und daß sich die Rechengeschwindigkeit verringert. Es muß jedoch festgehalten werden, daß einige Probleme nur in Festkommadarstellung sinnvoll gerechnet werden können.

Das trifft besonders für kommerzielle Rechnungen zu. Daher haben viele Automaten, wie z.B. der C 8205, meist die Möglichkeit, in beiden Zahlen-

darstellungen zu rechnen.

Sehr erwünscht ist es, daß man nur mit ganzen Zahlen rechnet. Das läßt sich auch mit der Festkommadarstellung erreichen, bei der ein Zahlenbereich

$$0 \leq |z| \leq (0,\ 777\ 777\ 777\ 77)_8$$

vorgegeben ist. Im Automaten wird ja lediglich die Ziffernfolge nach dem Komma verwandt. Die kleinste Zahl kann dabei als ganze Zahl angesehen werden.

$$0,000\ 000\ 000\ 01 \mathrel{\hat{=}} \ldots\ \ldots\ \ldots\ .1$$

Beim Rechnen ergeben sich für die Addition und Subtraktion keine Schwierigkeiten, denn es gilt:

$$\begin{array}{r} + \ldots\ \ldots\ \ldots\ .1 \\ + \ldots\ \ldots\ \ldots\ .1 \\ \hline \ldots\ \ldots\ \ldots\ .2 \end{array}$$

da der Automat rechnet

$$\begin{array}{r} 0,000\ 000\ 000\ 01 \\ + 0,000\ 000\ 000\ 01 \\ \hline 0,000\ 000\ 000\ 02 \end{array}$$

Anders hingegen bei der Multiplikation und Division. Hier liefert der Automat

$$\ldots\ \ldots\ \ldots\ .1 \times \ldots\ \ldots\ \ldots\ .1 = \ldots\ \ldots\ \ldots\ .0$$

denn es wird gerechnet

$$0,000\ 000\ 000\ 01 \times 0,000\ 000\ 000\ 01 =$$
$$0,000\ 000\ 000\ 000\ 000\ 000\ 1$$

Diesen Mangel kann man einmal damit beseitigen, daß man eine Verschiebeoperation anwendet.

Soll gerechnet werden:

0,000 000 000 01 x 0,000 000 000 01 setzt man

eine Verschiebung wie folgt an:

0,000 001 x 0,000 01 und das liefert 0,000 000 000 01

Ähnlich kann durch geschicktes Umformen auch eine ganzzahlige Division erzielt werden, worauf hier aber nicht eingegangen werden soll.

Für den Cellatron C 8205 gibt es Unterprogramme, die die ganzzahlige Multiplikation und Division ermöglichen.

## .3. Algorithmen

Neben der Kenntnis der spezifischen Darstellung der Information in der Datenverarbeitung sind Kenntnisse der Arbeitsweise erforderlich; insbesondere, wenn der sogenannte formallogische Teil der Arbeit durch Rechenautomaten ausgeführt werden soll. Die Erfassung der Arbeitsweise erfolgt über Algorithmen.

Der Algorithmus ist ein zentraler Begriff der Mathematik, der insbesondere in der maschinellen Rechentechnik große Bedeutung hat. Ohne auf seine Theorie eingehen zu wollen, soll er hier wie folgt charakterisiert werden.

Ein Algorithmus ist eine Rechenvorschrift, die folgenden Anforderungen genügen muß:

1. sie muß für eine Klasse von Aufgaben gelten
2. sie muß eindeutig sein
3. sie muß allgemeinverständlich sein (das soll heißen, daß sie auch von Bearbeitern verstanden und anwendbar sein muß, die die Theorie selber nicht beherrschen) und
4. sie muß endlich sein (das soll heißen, daß sie bei richtiger Vorgabe der Ausgangswerte und

logisch richtiger Anwendung zwangsläufig nach endlich vielen Schritten zum richtigen Ergebnis führt).

Algorithmen treten in unterschiedlichster Form auf. Wir finden sie als arithmetische Formeln zur Lösung numerischer Probleme. Beispielsweise ist für die Lösung der quadratischen Gleichung

$$ax^2 + bx + c = 0$$

der Algorithmus durch die bekannte Formel

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

gegeben. Dabei setzt man voraus, daß der Bearbeiter die Quadratwurzel aus einer Zahl ziehen kann. Das ist keineswegs selbstverständlich. Ein Rechenautomat kann dieses Problem im allgemeinen nicht unmittelbar lösen. Er kann die Quadratwurzel nur wiederum über einen Algorithmus lösen, Beispielsweise mittels des Algorithmus von Newton. Für jeden Radikanden $0 \leq R$ und für jede Ausgangsnäherung $u_0 > 0$ konvergiert

$$\frac{1}{2}\left(u_{n-1} + \frac{R}{u_{n-1}}\right) = u_n$$

gegen den Wert $\sqrt{R}$

Ist $\sqrt{2} = 1,41421 \ldots$ zu berechnen, so kann man mit $u_0 = 2$ als Ausgangsnäherung beginnen und erhält nach drei Schritten bereits eine Näherung, die auf vier Stellen nach dem Komma genau ist:

$$\frac{1}{2}\left(2 + \frac{2}{2}\right) = \frac{3}{2} = 1,5 = u_1$$

$$\frac{1}{2}\left(\frac{3}{2} + \frac{4}{3}\right) = \frac{17}{12} = 1,4166 = u_2$$

$$\frac{1}{2}\left(\frac{17}{12} + \frac{24}{17}\right) = \frac{577}{408} = 1,414216 = u_3$$

Der zuletzt angeführte Algorithmus ist für die Rechentechnik besonders interessant. Er demonstriert ein Iterationsverfahren. Bei der Iteration ist der Algorithmus so aufgebaut, daß eine Näherungslösung unter Einhalten eventuell vorgegebener Forderungen bei jedem Durchlauf verbessert wird. Dann kann man die Berechnung also mit einer recht groben Ausgangslösung so weit verbessern, wie es die verlangte Genauigkeit erfordert.

Ein Algorithmus kann aber auch durch eine Folge verbaler Anweisungen angegeben werden, was am Beispiel des Newton'schen Algorithmus demonstriert werden soll:

| Nr. | Instruktion | Es folgt |
|---|---|---|
| 1 | Eingabe R - Radikand<br>$\varepsilon$ - Genauigkeitsmaß<br>Es muß dabei gelten $0 < R$ | 2 |
| 2 | Man setze $u_0 = R$ ; $u_i := u_0$ | 3 |
| 3 | $u_{i-1} := u_i$ | 4 |
| 4 | $u_i = \frac{1}{2}\left(u_{i-1} + \frac{R}{u_{i-1}}\right)$ | 5 |
| 5 | Probe $\left\lvert u_{i-1} - u_i \right\rvert - \varepsilon < 0$?<br>falls ja<br>falls nein | <br>6<br>3 |
| 6 | Es ist $u_i \approx R$ | HALT |

In der Rechentechnik muß man jedoch an die Darstellungsform des Algorithmus höhere Anforderungen stellen. Hier muß der organisatorische und zeitliche Ablauf mit in die Darstellung einbezogen werden.

Es sei ergänzend bemerkt, daß ein Algorithmus die Arbeit mit konkreten numerischen Werten beschreibt. Er gibt an, wie aus Werten ( die durch Bezeichnungen fixiert werden), neue Werte zu berechnen sind (die dann wiederum zur Markierung eine Bezeichnung erhalten). In der Rechentechnik erfolgt diese Bezeichnung vielfach durch Adressen (Angabe des Platzes im Speicher, auf dem der Wert steht), oder es werden entsprechend der Bezeichnung Adressen zugeordnet. Alle genannten Forderungen lassen sich durch Plangleichungen erfüllen.

Plangleichungen werden vielfältig benutzt. Sie ermöglichen es, den zeitlichen Ablauf eines Rechen-, Zuordnungs- oder Speicherprozesses mit zu erfassen. Dies sei am Beispiel erklärt. Die übliche Gleichung

$$c = a + b$$

verlangt, daß zum fixierten Zeitpunkt t beide Seiten der Gleichung wertemäßig gleich sein sollen. Die Plangleichung verwendet das "Ergibt-Zeichen" :=, wobei der Doppelpunkt eine Pfeilspitze ersetzen soll. Die oben angeführte Gleichung als Plangleichung

$$c := a + b$$

würde bedeuten, daß zum Zeitpunkt t den mit a und b bezeichneten Größen ein Wert zugeordnet ist. Diese beiden Werte sollen nach der Rechen-Vorschrift + (Addition) verknüpft werden, wodurch (quasi zum Zeitpunkt t + 1) ein neuer Wert entsteht, der die Bezeichnung c erhält. Damit wird auch die Plangleichung

$$a := a + b$$

verständlich. Hier soll der errechnete Wert zum Zeitpunkt t + 1 die Bezeichnung a erhalten, wobei der alte Wert von a (der zum Zeitpunkt t vorlag)

verloren geht. Die Gleichung a = a + b würde dagegen zur Folge haben, daß gelten müßte, b = 0. Der geschilderte Sachverhalt ist für die Rechentechnik von besonderer Bedeutung, wenn als Bezeichnungen . Adressen verwendet werden. Gilt für den Inhalt einer Adresse das Symbol $\langle \ldots \rangle$ , so sagt die Plangleichung

$$4000 := 4000 + 3000$$

der Inhalt der Zellen (Speicherstellen für Werte) mit der Adresse 3000 und 4000 sollen (zum Zeitpunkt t) addiert werden. Den Wert des Ergebnisses soll man dann (zum Zeitpunkt t + 1) wieder in die Zelle mit der Adresse 4000 zurücktransportieren.

Allgemein kann man sagen, durch die Plangleichung .

$$y := f(x_1, x_2, x_3, \ldots, x_n)$$

bringt man zum Ausdruck, daß die Werte der durch $x_1, x_2, \ldots, x_n$ bezeichneten Größen entsprechend der Vorschrift $f$ zu neuen Werten verknüpft werden sollen, die man mit y bezeichnet. Als Beispiel werde die Vorschrift für die skalare Multiplikation angeführt. Sei

$\mathfrak{a} = (a_1, a_2, a_3)$ und $\mathfrak{b} = (b_1, b_2, b_3)$ gegeben, so ist

$$\mathfrak{a} \cdot \mathfrak{b} = (a_1, a_2, a_3) \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$$

das skalare Produkt.

Für die Berechnung schreibt man, wenn noch $s_0 = 0$ als Ausgangssumme gesetzt wird:

$$s_i = s_{i-1} + a_i \cdot b_i \quad (i = 1, 2, 3)$$

oder unter Verwendung der Plangleichung

$$s := s + a \cdot b,$$

wobei allerdings für die organisatorischen Anweisungen zusätzliche Vereinbarungen zu treffen sind.

Die letzte Darstellung ermöglicht mit s als Zwischensumme folgende Berechnung auszuführen:

0. Schritt $s_0 := 0$
1. Schritt $s_1 := s_0 + a_1 \cdot b_1$
2. Schritt $s_2 := s_1 + a_2 \cdot b_2$
3. Schritt $s_3 := s_2 + a_3 \cdot b_3$

$S_3$ liefert dann das Ergebnis, das skalare Produkt. Auch für folgende Festlegungen verwendet man Plangleichungen:

n := 1
s := 0
n := n + 1

In den ersten beiden Fällen werden vorgegebene Werte einer Benennung zugeordnet. Damit wird praktisch die Definition des Wertes einer Größe vorgenommen. Im letzteren wird der Wert der Größe n um 1 erhöht und dann wieder n zugeordnet, wodurch das Anwachsen eines Index leicht ausgedrückt werden kann.

Es seien nunmehr die zusätzlichen organisatorischen Forderungen angedeutet. Der Algorithmus soll für die Rechentechnik so aufbereitet sein, daß sein gesamter Ablauf eindeutig erkennbar ist. Hierzu bedient man sich unterschiedlichster Mittel. Besonders geeignet ist das Flußdiagramm, auch Flußbild genannt.

Das Flußbild ist eine detaillierte grafische Darstellung eines Algorithmus. Es stellt die Verbindung zwischen dem mathematischen Modell mit seinen Lösungsalgorithmen und dem späteren Maschinenprogramm dar.

Zur grafischen Darstellung für Flußdiagramme wurde von John v. Neumann die Kästchenmethode vorgeschlagen. Alle Formeln, Anweisungen und sonstigen Festlegungen werden in Kästchen eingeschlossen und der Gesamtfluß durch Flußlinien erfaßt. Diese Art erfordert jedoch einen hohen Zeichenaufwand und ist bei Algorithmen relativ unübersichtlich. Daher hat sich in den letz-

ten Jahren immer stärker die Leitlinienmethode durchgesetzt. Der Entwurf des DDR-Standards "Sinnbilder für Datenfluß- und Programmablaufpläne" führt die Symbole beider Methoden gleichberechtigt an.

Da der Leitlinienmethode der Vorrang eingeräumt werden sollte, wird sie auf der Grundlage der im oben erwähnten Standard angewiesenen Symbole demonstriert.

Jeder (Haupt-) Algorithmus beginnt mit einem quadratischen Kästchen START und endet mit einem gleichen Kästchen STOPP. Diese Kästchen nennt man Organisationskästchen. Bei Nebenalgorithmen [1]) sind diese Organisationskästchen Kreise, in denen "B" statt "START" und "E" statt "STOPP" eingetragen werden kann (Abb. 2).

Beide Organisationskästchen werden durch eine größtenteils senkrecht verlaufende Leitlinie verbunden, wie es Abb. 2 zeigt. Waagerechter Verlauf der Leitlinie ist nur zur Verbindung erlaubt, schräg zur Leitlinie verlaufende Verbindungen sind verboten.

Alle Informationen, Weisungen und Texte sind am senkrechten Teil rechts neben der Leitlinie anzubringen. Kommentare (Informationen) können an beliebiger Stelle des Flußbildes eingeschoben werden. Sie werden durch einen rechts neben der Leitlinie angebrachten Parallelstrich gekennzeichnet (s. Abb. 2).

Kommentare sind grundsätzlich nach dem Organisationskästchen START zu geben. Sie müssen dort enthalten:

---

[1]) Nebenalgorithmen sind häufig wiederkehrende Rechenvorschriften, die im übergeordneten Hauptalgorithmus gegebenenfalls mehrfach vorkommen können. Hauptalgorithmen entsprechen den späteren Hauptprogrammen und Nebenalgorithmen den Unterprogrammen.

- bei Hauptalgorithmen
  Name des Algorithmus,
  formale Parameter des Input,
  Eingabewerte (Input),
  Regieanweisungen (gegenüber Einschränkungen),

- bei Nebenalgorithmen
  Name des Algorithmus,
  formale Parameter des Input,
  Regieanweisungen für den Einbau in einen Hauptalgorithmus

Auch vor STOPP und [illegible] muß ein Kommentar gegeben werden. Er soll enthalten:

- bei Hauptalgorithmen
  Ausgabewerte (Output),

- bei Nebenalgorithmen
  formale Parameter des Output,
  Regieanweisungen über den Nicksprung in den Hauptalgorithmus.

Die geforderten Angaben können beliebig ergänzt werden. Desgleichen können zusätzliche Kommentare beliebigen Inhaltes an jeder senkrechten Stelle der Leitlinie eingefügt werden. Man sollte sich jedoch auf wesentliche Anmerkungen beschränken, da durch zu viele Kommentare die Übersicht leiden kann.

Weisungen werden durch Plangleichungen ausgedrückt. Sie sind rechts neben den senkrechten Teil der Leitlinie zu schreiben, um in der Leitlinie durch einen waagerechten Querstrich markiert zu werden (Abb. 1). Weisungen dürfen nicht in Kommentare eingebaut werden. Ein Querstrich durch den senkrechten Doppelstrich eines Kommentares ist also unzulässig.

Jeder Test ist als Alternativtest auszuweisen. Die Markierung auf der Leitlinie erfolgt wie bei Weisungen. Der Test wird wie folgt dargestellt. Ein großes P kennzeichnet den Test. Darauf folgt die

Alternative durch ein Fragezeichen abgeschlossen, in einer geschweiften Klammer, also P {Alternative?}.

Nach jedem Test muß, allerdings nicht unmittelbar, eine Verzweigung des Algorithmus erfolgen. Die Verzweigung wird auf der Leitlinie durch einen Punkt angegeben, von dem aus ein waagerechter Zweig der Leitlinie nach links oder nach rechts abgeht. Die linke Verzweigung ist nur möglich, wenn es sich um einen einfachen Vor- oder Rücksprung im Algorithmus handelt. Bei diesem Zweig darf keine Weisung und kein Test auftreten! Nach der Verzweigung muß jeder Zweig markiert werden, und zwar mit

j oder P, wenn der Zweig bei positivem Ausfall des Alternativtests weiterbearbeitet werden soll,

n oder $\overline{P}$, wenn die Weiterarbeit bei negativem Ausfall des Tests erfolgen soll.

Erhält der Nebenwert mindestens eine Weisung oder einen Test, muß er nach rechts waagerecht ausgeführt werden. Weisungen und Teste dürfen aber erst nach rechtwinkligem Knick der Nebenlinie in einem senkrechten Verlauf (parallel zum Hauptzweig) angebracht werden. Dabei ist es gleichgültig, ob es sich um einen Rück- oder Vorwärtssprung handelt. Die Rückführung eines Nebenzweiges erfolgt durch eine Pfeilspitze ohne Markierung auf der Leitlinie, in die zurückgeführt wird. Leitlinien sollten sich nicht kreuzen.
Will man die Darstellung an Algorithmen unterbrechen, verwendet man feste Konnektoren (Abb. 3). Dabei können mehrere Absprungstellen zu einer Einsprungstelle führen. Zum Konnektor können noch Hinweise zum Auffinden des Abschlusses hinzugefügt werden.

Für mehrfache Verzweigungen nach festgelegten Bedingungen verwendet man variable Konnektoren (Abb. 4). Der Einsprung richtet sich nach dem aktuellen Wert

des Parameters

$$A_i = \{A_1, A_2, A_3\}$$

Schließlich kann noch ein Nebenalgorithmus wie ein Unterprogramm oder ein Programmteil eingefügt werden, wie es Abb. 5 zeigt. Hier muß durch vor- und nachgeordnete Kommentare der Anschluß an den Hauptalgorithmus gesichert werden.

Die Darstellung der Algorithmen als Flußdiagramm soll an Beispielen demonstriert werden.

Beispiel 1

Es soll ein lineares Gleichungssystem von zwei Gleichungen mit zwei Unbekannten gelöst werden. Mit Hilfe des bekannten Additionsverfahrens erhält man

$$\begin{array}{lll} a_{11}\,x_1 + a_{12}\,x_2 = b_1 & a_{22} & -a_{21} \\ a_{21}\,x_1 + a_{22}\,x_2 = b_2 & -a_{12} & a_{11} \end{array}$$

$$\begin{array}{l} a_{11}\,a_{22}\,x_1 + a_{12}\,a_{22}\,x_2 = b_1\,a_{22} \\ -a_{12}\,a_{21}\,x_1 - a_{12}\,a_{22}\,x_2 = -b_2\,a_{12} \end{array}$$

$$(a_{11}\,a_{22} - a_{12}\,a_{21})\,x_1 = b_1\,a_{22} - b_2\,a_{12}$$

$$x_1 = \frac{b_1\,a_{22} - b_2\,a_{12}}{a_{11}a_{22} - a_{12}a_{21}}$$

$$\begin{array}{l} -a_{11}\,a_{21}\,x_1 - a_{12}\,a_{21}\,x_2 = -b_1\,a_{21} \\ a_{11}\,a_{21}\,x_1 + a_{11}\,a_{22}\,x_2 = +b_2\,a_{11} \end{array}$$

$$(a_{11}\,a_{22} - a_{12}\,a_{21})\,x_2 = b_2\,a_{11} - b_1\,a_{21}$$

$$x_2 = \frac{b_2\,a_{11} - b_1\,a_{21}}{a_{11}a_{22} - a_{12}a_{21}}$$

Die hierbei auftretenden Ausdrücke

$$a_{11}\,a_{22} - a_{12}\,a_{21}$$

$$b_1 a_{22} - b_2 a_{12}$$
$$b_2 a_{11} - b_1 a_{21}$$

nennt man Determinanten (2. Ordnung), die nach folgender Vorschrift geschrieben und berechnet werden:

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} a_{22} - a_{12} a_{21} = \Delta$$

$$\begin{vmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{vmatrix} = b_1 a_{22} - b_2 a_{12} = \Delta_1$$

$$\begin{vmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{vmatrix} = b_2 a_{11} - b_1 a_{21} = \Delta_2$$

Sichert man, daß stets gilt $\Delta \neq 0$, so ist mit

$$\frac{\Delta_1}{\Delta} = x_1$$
$$\frac{\Delta_2}{\Delta} = x_2$$

ein Algorithmus gegeben, wie Abb. 6 darstellt.

Beispiel 2

Beim vorhin betrachteten Beispiel soll nicht vorausgesetzt werden, daß $\Delta \neq 0$ ist. Dann muß das Flußbild der Abb. 6 erweitert werden und man erhält einen verzweigten Algorithmus (Abb. 7).
Nach Berechnung von

$$a_{11} a_{22} - a_{12} a_{21} = \Delta$$

fügt man in den Text eine Alternative ein

$$\Delta = 0?$$

Bei "JA", also $\Delta = 0$, kann nicht weitergerechnet

werden, da dann $\frac{\Delta 1}{\Delta}$ nicht definiert ist. Hier liegt ein Sonderfall vor, den man durch "STOPP" kennzeichnet.

Bei "NEIN", also $\Delta \neq 0$, verläuft die Rechnung wie zuvor. Allerdings handelt es sich hier nur um ein Modellbeispiel, denn praktisch werden lineare Gleichungssysteme nicht nach dieser Methode berechnet.

Beispiel 3

Das Produkt der ersten n Zahlen wird als Fakultät bezeichnet und wie folgt beschrieben:

$$1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots \cdot (n-1) \cdot n = n!$$

(sprich "n Fakultät").

Speziell ist

$$\begin{aligned}
1! &= 1 = 1\\
2! &= 1 \cdot 2 = 2\\
3! &= 1 \cdot 2 \cdot 3 = 6\\
4! &= 1 \cdot 2 \cdot 3 \cdot 4 = 24\\
5! &= 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120\\
6! &= 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720\\
7! &= 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 = 5040\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots
\end{aligned}$$

Die Berechnung dieser Funktion erfolgt über die Plangleichungen

$$\begin{aligned}
k &:= 1\\
i &:= 1\\
i &:= i \cdot k\\
p &:= k - n\\
k &:= k + 1
\end{aligned}$$

wobei die Größe p als Prüfgröße eingeführt werden muß. Ist $o < p$ folgt, daß auch gilt $k < n$. Dann entspricht das Ergebnis noch nicht dem verlangten Wert, und die Berechnung ist nach Erhöhen von $k := k + 1$ zu wiederholen. Erst bei $0 = p$ gilt

k = n und der verlangte Wert liegt vor. Abb. 8 zeigt das zugehörige Flußdiagramm.

Beim Aufstellen des Flußbildes für einen Algorithmus können Fehler unterlaufen. Daher sollten auch vielfältigste Kontrollen in den gesamten Bearbeitungsprozeß eingebaut werden. Die erste Kontrolle sollte am Flußbild erfolgen.

Zur Durchführung der Kontrolle werden alle im Flußbild angeführten Größen in einer Tabelle erfaßt und darauf streng nach dem Ablauf des Flußbildes für ein nicht triviales aber übersehbares Beispiel eine manuelle Rechnung durchgeführt. Das oder die Beispiele müssen dabei so gewählt werden, daß alle Teste mit jeder Entscheidung mindestens einmal angesprochen werden. Man muß also alle Schleifen und Algorithmusteile mindestens einmal durchlaufen. Dies werde am Beispiel 3, Abb. 8 für n = 4, also n! = 24 demonstriert:

| Input | k | i | p | P p 0 ? | Output n! |
|---|---|---|---|---|---|
| 4 | 0 | 1 | | | |
| | 1 | 1 | -3 | ja | |
| | 2 | 2 | -2 | ja | |
| | 3 | 6 | -1 | ja | |
| | 4 | 24 | 0 | nein | 24 |

Wie ersichtlich, wurde das Flußdiagramm richtig dargestellt. Für ein derart einfaches Beispiel ist dies leicht zu übersehen. Der Vorteil einer solchen Kontrolle macht sich jedoch erst bei komplizierteren Beispielen mit vielen Verzweigungen bemerkbar, wenn dabei der Rechenaufwand auch beträchtlich sein kann. Dieser Aufwand ist meist weitaus geringer als eine Fehlerkorrektur beim Erproben eines nach dem (falschen) Flußbild dargestellten Programmes.

Als nächstes Beispiel werde die Berechnung einer Quadratwurzel besprochen (Abb. 84).

Beispiel 4

Die Berechnung der Quadratwurzel $\sqrt{R}$ kann bekanntlich nach der Plangleichung

$$\frac{1}{2}\left(u + \frac{R}{u}u\right)$$

ausgeführt werden kann.

Das Verfahren konvergiert, wie bereits gesagt, für jeden Radikanden $0 < R$ und jede Anfangsnäherung $0 < u_0$. Die Genauigkeit der Näherung kann durch die absoluten Beträge der Differenz zweier aufeinanderfolgender Näherungen bestimmt werden. Dieser Betrag kann kleiner gemacht werden als eine vorher festgelegte Genauigkeitsschranke $\varepsilon$. Hier liefert die Kontrolle für das Beispiel R = 2:

| Input | | u | u' | p | P p < 0? | Output |
|---|---|---|---|---|---|---|
| R+1 | R+2 | | | | | R+1 |
| R=2 | $\varepsilon$=0,01 | 2 | 1,5 | 0,49 | nein | |
| | | 1,5 | 1,4166 | 0,07 | nein | |
| | | 1,4166 | 1,414216 | -0,008 | ja | 1,414 |

2. Zahlenumwandlung

Nunmehr sind die Voraussetzungen gegeben, um Zahlen aus einem Zahlensystem in ein anderes umzurechnen.

Die Zahlenumwandlung aus dem Oktal- in das Dualsystem ist außerordentlich einfach durch die Tatsache, daß die Basiszahl des Oktalsystems eine Potenz des Dualsystems ist

$$8 = 2^3,$$

worauf bereits hingewiesen wurde. Man kann Oktalzahlen ziffernweise unter Beibehalten der Position in eine Dualtriade umwandeln.

Dualzahlen brauchen lediglich vom Komma beginnend nach Dualtriaden aufgeteilt und ergänzt zu werden, und es kann jede Dualtriade direkt in eine Oktalziffer umgerechnet werden.

Damit ist also, wenn jeweils die Position beibehalten wird, die Umwandlung der gesamten Zahl vollzogen. Dabei entsprechen einander:

| oktal | dual | oktal | dual |
|---|---|---|---|
| 0 | 000 | 4 | LOO |
| 1 | 00L | 5 | LOL |
| 2 | 0LO | 6 | LLO |
| 3 | 0LL | 7 | LLL |

Beispiel: $(4307,216)_8$ = LOO OLL 000 LLL, OLO 00L LLO

$(123,24)_8$=00L 0LO 0LL, 0LO LOO = L 0LO 0LL, 0LO L

LOO'OLL'00[illegible], 0LO'00L'LLO =

LOO'OLL'00L LLL, 0LO'00L'LLO = $(4307,216)_8$

L'0LO'0LL,0LO'L = 00L'0LO'0LL,0LO'LOO = $(123,24)_8$

Da die Basiszahl des Dezimalsystems keine Potenz eines

der anderen hier interessierenden Zahlensysteme ist, sind etwas kompliziertere Algorithmen heranzuziehen.

Es werde zuerst das Umrechnen einer Dezimalzahl in eine Oktalzahl betrachtet. Dieser Prozeß wird Konvertieren genannt. Rekonvertieren heißt dann das Umwandeln einer Dualzahl in eine Dezimalzahl. Für diese Rechnungen werden meist die Rechenautomaten selber herangezogen. Dennoch kommt es vor, daß auch der Programmierer oder der Bedienungstechniker eine Zahl konvertieren oder rekonvertieren muß.

Die nachfolgend angeführte einfache Umwandlung beruht auf der mathematischen Disziplin des Rechnens mit Restklassen. Dabei soll zur Vereinfachung der ganzzahlige Anteil einer Dezimalzahl vom gebrochenen Teil unterschieden werden.

Der ganzzahlige duale Anteil einer Dezimalzahl kann durch eine wiederholte Division des Zahlenteiles durch zwei mit Rest ermittelt werden. Der Rest kann dabei ja nur 1 oder 0 sein und liefert jeweils der Reihe nach eine Dualstelle mit der kleinsten beginnend.

Beispiel: 53 soll konvertiert werden

53 : 2 = 26 Rest 1
26 : 2 = 13 Rest 0
13 : 2 = 6 Rest 1
6 : 2 = 3 Rest 0
3 : 2 = 1 Rest 1
1 : 2 = 0 Rest 1

Somit ist die konvertierte Zahl:

53 = LLOLOL

Die Probe erhält man durch Rekonvertieren. Hierzu verwendet man die dem Heft beigefügte Potenztabelle. Man zählt die Exponenten ab und addiert für jedes L den entsprechenden Wert der Dualpotenz.

LLOLOL ergibt somit:

$$1 \cdot 2^0 = 1$$
$$0 \cdot 2^1 = 0$$
$$1 \cdot 2^2 = 4$$
$$0 \cdot 2^3 = 0$$
$$1 \cdot 2^4 = 16$$
$$1 \cdot 2^5 = 32$$

$$1 + 4 + 16 + 32 = 53$$

Somit gilt

LLOLOL = 53.

Sehr einfach ist das Konvertieren von Dezimalbrüchen. Nach dem unten angeführten Schema ist der Dezimalbruch lediglich jeweils mit zwei zu multiplizieren und die in den "Überlauf" einfließende Null oder Eins in Dualziffern zu schreiben.

Beispiel: Z = 0,171875 ist zu konvertieren

| | |
|---|---|
| 0 | 171875 . 2 |
| 0 | 343750 . 2 |
| 0 | 687500 . 2 |
| 1 | 375000 . 2 |
| 0 | 750000 . 2 |
| 1 | 500000 . 2 |
| 1 | 000000 . 2 |

Die gesuchte Dualzahl ist gleich der Ziffernfolge in der linken Spalte, wenn nach der ersten Null das Komma und alle Ziffern in das Dualsystem gesetzt werden.

Z = 0, 00LOLL

Die linke Spalte ist bei der Multiplikation nicht zu berücksichtigen. Die Probe kann man wiederum durch Rekonvertieren vornehmen. Vergleicht man die Stellenwerte der Dualzahlen, so ergibt sich:

$0 \cdot 2^{0} = 0$
$0 \cdot 2^{-1} = 0$
$0 \cdot 2^{-2} = 0$
$1 \cdot 2^{-3} = 0,125$ 0,125
$0 \cdot 2^{-4} = 0$ + 0,03125
$1 \cdot 2^{-5} = 0,03125$ + 0,015625
$1 \cdot 2^{-6} = 0,015625$ 0,171875

Ist eine Zahl gegeben, die aus einem ganzzahligen Anteil und aus einem Dezimalbruch besteht, beispielsweise Z = 321, 34375, so kann erst der ganzzahlige und anschließend der gebrochene Teil konvertiert werden.

Beispiel: Z = 321,34375

I.
| | | | |
|---|---|---|---|
| 321 | : 2 = | 160 | Rest 1 |
| 160 | : 2 = | 80 | Rest 0 |
| 80 | : 2 = | 40 | Rest 0 |
| 40 | : 2 = | 20 | Rest 0 |
| 20 | : 2 = | 10 | Rest 0 |
| 10 | : 2 = | 5 | Rest 0 |
| 5 | : 2 = | 2 | Rest 1 |
| 2 | : 2 = | 1 | Rest 0 |
| 1 | : 2 = | 0 | Rest 1 |

Der ganzzahlige Anteil ist somit LOLOOOOL.

II.
| | |
|---|---|
| 0 | 34375 . 2 |
| 0 | 68750 . 2 |
| 1 | 37500 . 2 |
| 0 | 75000 . 2 |
| 1 | 50000 . 2 |
| 1 | 00000 . 2 |

Der Bruchteil ist 0,OLOLL.
Die gesamte Zahl ergibt sich als:

321,34375 = L OL 00 00 OL, OL OL L

Zur Probe der Rekonvertierung:

| | | |
|---|---|---|
| $1 \cdot 2^{8}$ | = 256,0 | |
| $0 \cdot 2^{7}$ | = 0,0 | |
| $1 \cdot 2^{6}$ | = 64,0 | |
| $0 \cdot 2^{5}$ | = 0,0 | |
| $0 \cdot 2^{4}$ | = 0,0 | |
| $0 \cdot 2^{3}$ | = 0,0 | |
| $0 \cdot 2^{2}$ | = 0,0 | 256,0 |
| $0 \cdot 2^{1}$ | = 0,0 | 64,0 |
| $1 \cdot 2^{0}$ | = 1,0 | 1,0 |
| $0 \cdot 2^{-1}$ | = 0,0 | 0,25 |
| $1 \cdot 2^{-2}$ | = 0,25 | 0,0625 |
| $0 \cdot 2^{-3}$ | = 0,0 | 0,03125 |
| $1 \cdot 2^{-4}$ | = 0,0625 | 321,34375 |
| $1 \cdot 2^{-5}$ | = 0,03125 | |

Für das Rekonvertieren gilt somit grundsätzlich, daß nach dem Zahlenaufbau (5) die Produkte aus Ziffernwert und Stellenwert zu summieren sind. Beim Dualsystem vereinfacht sich dies dadurch, daß lediglich die Ziffernwerte 0 und 1 auftreten können. So braucht man nur entsprechend den mit L versehenen Positionen der Zahl die Stellenwerte zu addieren. Hierzu hilft die beigefügte Tabelle 1.

Konvertieren und Rekonvertieren liefert jedoch nicht für jeden endlichen Dezimalbruch auch einen endlichen Dualbruch. Das führt bei begrenzter Stellenzahl im Automaten zu Konvertierungs- respektive Rekonvertierungsfehlern.

Zur besseren Übersicht sei angenommen, daß für den Automaten 4 Dezimalstellen und 9 Dualstellen nach dem Komma vorgesehen sind. Dann würde sich 0,6250 ohne Fehler in 0, LOL 000 000 konvertieren und rekonvertieren lassen, denn es ist

| | | | |
|---|---|---|---|
| 0 | 6250 | und | $1 \cdot 2^{-1} = 0,5$ |
| 1 | 2500 | | $0 \cdot 2^{-2} = 0$ |
| 0 | 5000 | | $1 \cdot 2^{-3} = 0,125$ |
| 1 | 0 | | 0,625 |

Aber bereits für 0,1000 würde sich ein spürbarer Konvertierungsfehler ergeben, denn es ist

| 0 | 1000 |
|---|---|
| 0 | 2 |
| 0 | 4 |
| 0 | 8 |
| 1 | 6 |
| 1 | 2 |
| 0 | 4 |
| 0 | 8 |
| 1 | 6 |
| 1 | 2 |

also 0,1000 $\hat{=}$ 0, 000 LLO OLL entsprechend der Stellenzahlen des Automaten. Das Rekonvertieren ergibt jedoch

$$1 \cdot 2^{-4} = 0{,}062\ 5$$
$$1 \cdot 2^{-5} = 0{,}031\ 25$$
$$1 \cdot 2^{-8} = 0{,}003\ 906\ 25$$
$$1 \cdot 2^{-9} = 0{,}001\ 953\ 125$$
$$0{,}099\ 609\ 375$$

der möglichen Stellenzahl entsprechend, also

0,0996

somit einen Fehler der Größe 0,0004. Derartige Fehler können sich insbesondere beim Test auswirken, wenn beispielsweise eine vorgegebene Größe 0,1 mit einer errechneten Größe verglichen werden soll. Es kann dabei auftreten, daß durch Konvertierungs- und Rekonvertierungsfehler die Differenz zweier ursprüngl gleicher Größen einen Wert ungleich Null ergibt. Dies ist bei der Arbeit mit Automaten zu berücksichtigen.

Das Umwandeln aus dem Dezimal- in das Oktalsystem und umgekehrt kann nach denselben Vorschriften er-

folgen. Dies soll am letzten Beispiel veranschaulicht werden.

Beispiel: 321, 343 75

Es ergibt sich zu

321 : 8 = 40 Rest 1
40 : 8 = 5 Rest 0
5 : 8 = 0 Rest 5

Der ganzzahlige Anteil ist $(501)_8$

| | |
|---|---|
| 0 | 34375 . 8 |
| 2 | 75000 . 8 |
| 6 | 00 |

Der Bruchteil ist $(0,26)_8$

| Probe: | | | | |
|---|---|---|---|---|
| 5 . | 64 | = 320,0 | 320,0 | . |
| 0 . | 8 | = 0,0 | 1,0 | |
| 1 . | 1 | = 1,0 | 0,25 | |
| 2 . | 0,125 | = 0,25 | 0,09375 | |
| 6 . | 0,015625 | = 0,09375 | 321,34375 | |